

# **ASP+ Tips And Tricks**

**Scott Guthrie**  
**.NET Frameworks**  
**Microsoft Corporation**

**4-322**

Microsoft®  
**PDC** 2000  
Professional Developers Conference

the defining

***point***

# Agenda

- **Goal: Demonstrate a host of ASP+ tips and tricks for building great Web Apps**
  - Walkthrough instructions available for download on newsgroups & Post CD
- **Four Themes:**
  - Deploying Web Applications
  - Debugging Web Applications
  - Running Web Applications
  - Miscellaneous other stuff I think is

# Deploying Web Applications

# Deploying Web Applications

- Goal: Enable a web app to be deployed simply by copying its bits to the server
- No-nonsense architecture
  - No admin or registration utilities required
  - No local server access required
  - No web server restarts required
- Must work for all web app resources
  - Web Pages/ Web Services

# Improved Code Deployment

**Simple deployment for components**

- Just xcopy/ftp to an application's “\bin” dir
- No registration required (no more regsvr32)
- Each web app is isolated from other apps
  - Components can be private to application
  - Other apps cannot load private components



# Code Registration Steps

1. Compile code into a .Net Library
  - `csc.exe /target:library MyClass.cs`
  - `vbc.exe /target:library MyClass.vb`
2. Copy DLL into “bin” dir immediately under the application VRoot
  - `copy MyClass.dll \wwwroot\bin\`
3. Create and use class from any ASP+ page, service or other components

# Code Uninstall Steps

1. Delete Library from “bin”dir
  - `del MyClass.dll`



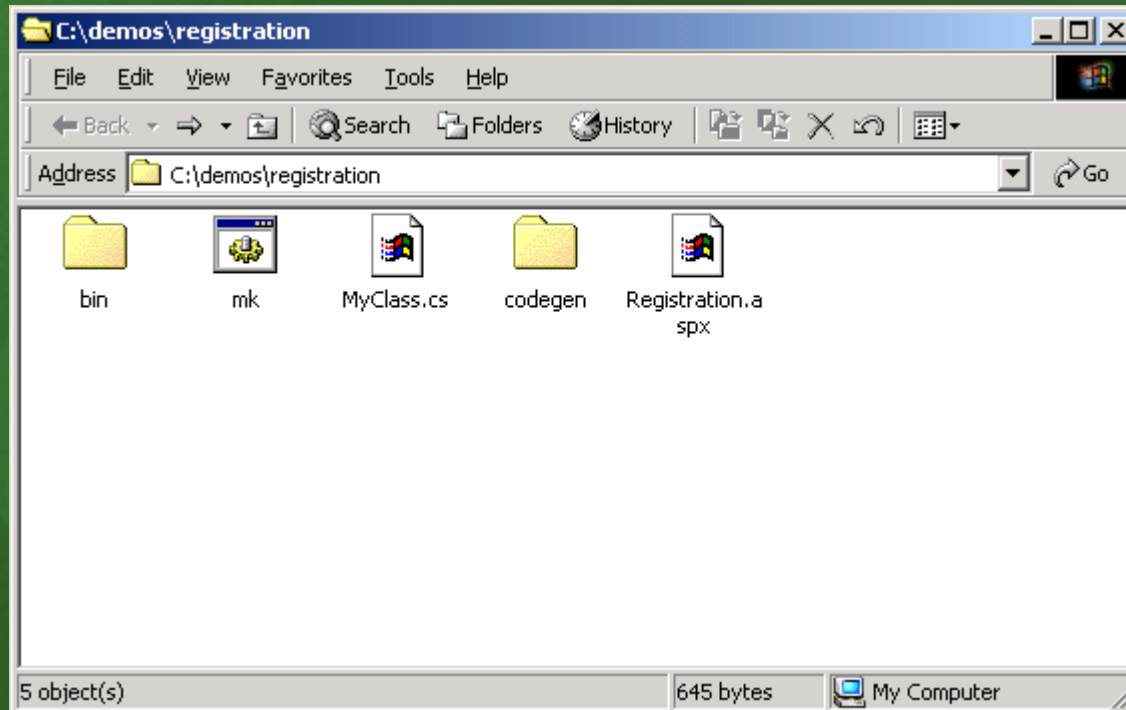
# Dynamic Code Update

- It is now possible to dynamically update compiled DLLs on a server
  - No more locked dlls
  - No admin tool required
  - No utility programs required
  - No configuration changes required
  - It just works....
- Just use xcopy, ftp, dav or frontpage to copy new DLL on top of old one

# Dynamic Code Update Steps

1. Deploy original .Net Library DLL into application's "bin" directory
2. Use .Net component somewhere within an ASP+ Application
3. Make a source change to the .Net Library DLL and recompile it
4. Copy it again into the app's "bin" directory - notice app starts using it

# Code Registration Demos



# XML Configuration

- Goal: Provide *extensible* configuration for *admins & developers* to *hierarchically* apply settings for an *application*
- Configuration data stored in XML text files
  - Format is human-readable and human-writable
- Config data can be stored in app directories
  - Config system automatically detects changes

# Session State Configuration

- Session State is config.web friendly
  - Timeouts
  - External State Store (more on later)
  - Cookieless session state tracking option

# **Cookieless Sessions**

- **Session State no longer requires client cookie support for SessionID**
  - **Can optionally track SessionID in URL**
- **Can require no code changes to app**
  - **All relative links continue to work**



# Cookieless Sessions

## Steps

1. Create “config.web” file in app vroot

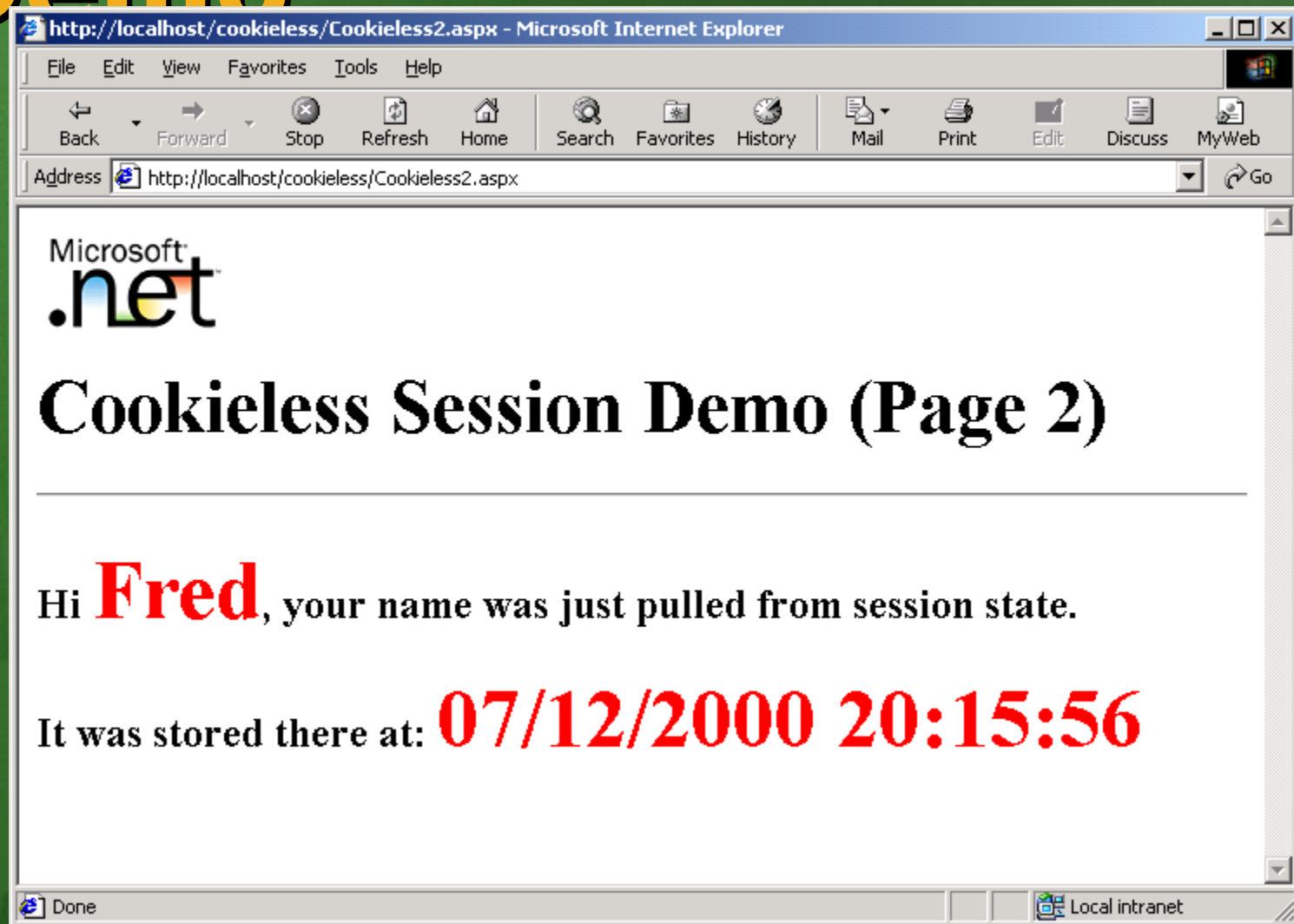
2. Add following text:

```
<configuration>
```

```
    <sessionstate cookieless=“true”/>
```

```
</configuration>
```

# Cookieless Session Demo



# App Settings

- **Application specific settings can also now be stored in config.web files**
  - **Enables devs to avoid hard-coding them**
  - **Administrators can later change them**
- **Examples:**
  - **Database DSN Settings**
  - **File Locations**

# App Settings Steps

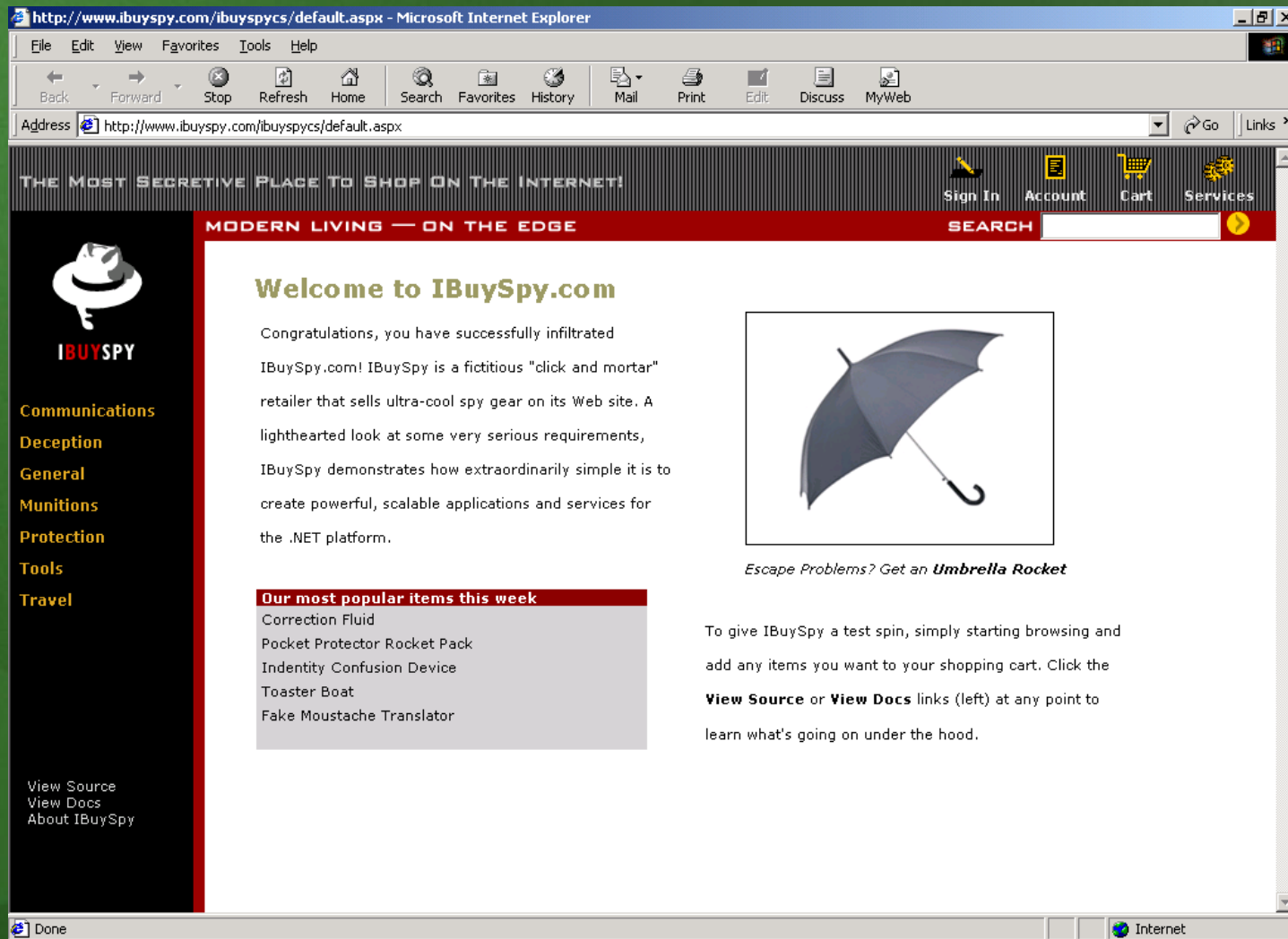
1. Create “config.web” file in app root:

```
<configuration>  
  <appsettings>  
    <add key=“dsn”  
value=“localhost;uid=sa;pwd=;Database=foo”/>  
  </appsettings>  
</configuration>
```

2. Below code returns hashtable of values:

- Dim Config as HashTable
- Config =  
Context.GetConfig(“appsettings”)

# App Settings Demo



# **Debugging Web Applications**



# Tracing

- **ASP+ supports page and app tracing**
  - Easy way to include “debug” statements
  - No more messy `Response.Write()` calls!
- **Great way to collect request details**
  - Server control tree
  - Server variables, headers, cookies
  - Form/QueryString parameters

# Page Tracing Steps

- 1. Add trace directive at top of page**
  - `<%@ Page Trace="True" %>`
- 2. Add trace calls throughout page**
  - 1. `Trace.Write("MyApp", "Button Clicked")`**
  - 2. `Trace.Warn("MyApp", "Value: " + value)`**
- 3. Access page from browser**

# Page Tracing Demo

http://localhost/debug/AspConf.aspx - Microsoft Internet Explorer - Daily Build 5.50.4030.1400

File Edit View Favorites Tools Help

Back Forward Stop Refresh Home Search Favorites History Mail Print Edit Discuss MyWeb

Address http://localhost/debug/AspConf.aspx Go Links »

### Request Details

<b>Session Id:</b>	seqqjtbqcjebclf20d1x4ra2	<b>Request Type:</b>	POST
<b>Time of Request:</b>	06/21/2000 14:22:56	<b>Status Code:</b>	200
<b>Compiled:</b>		<b>Cached:</b>	

### Trace Information

Category	Message	From First(s)	From Last(s)
aspx.page	Begin Init		
aspx.page	End Init	0.000000	0.000000
aspx.page	Begin LoadState	0.000000	0.000000
aspx.page	End LoadState	0.000000	0.000000
aspx.page	Begin ProcessPostData	0.000000	0.000000
aspx.page	End ProcessPostData	0.010015	0.010015
aspx.page	Begin PreRender	0.010015	0.000000
aspx.page	End PreRender	0.010015	0.000000
aspx.page	Begin SaveState	0.060088	0.050074
aspx.page	End SaveState	0.060088	0.000000
aspx.page	Begin Render	0.060088	0.000000
aspx.page	End Render	0.130191	0.070103

### Control Tree

Control Id	Type	Render Size bytes (including children)	Viewstate Size bytes (excluding children)
Page	ASP.AspConf.aspx	2025	0
ctrl0	System.Web.UI.LiteralControl	2	0
ctrl1	System.Web.UI.LiteralControl	93	0
ctrl2	System.Web.UI.LiteralControl	24	0
ctrl3	System.Web.UI.WebControls.AdRotator	---	0
ctrl4	System.Web.UI.LiteralControl	11	0
ctrl5	System.Web.UI.HtmlControls.HtmlForm	2043	0
ctrl6	System.Web.UI.LiteralControl	26	0
Name	System.Web.UI.WebControls.TextBox	---	0
ctrl7	System.Web.UI.LiteralControl	26	0
Category	System.Web.UI.WebControls.DropDownList	---	0

Done Local intranet

# App Tracing Steps

1. Create “config.web” file in app root:

```
<configuration>
```

```
    <trace enabled=“true” requestlimit=“10”/>
```

```
</configuration>
```

2. Access tracing URL within app
  - <http://localhost/approot/Trace.axd>

# App Tracing Demo

http://localhost/debug/trace.axd - Microsoft Internet Explorer - Daily Build 5.50.4030.1400

File Edit View Favorites Tools Help

Back Forward Stop Refresh Home Search Favorites History Mail Print Edit Discuss MyWeb

Address http://localhost/debug/trace.axd Go Links >>

## Application Trace

debug

[ [clear current trace](#) ]

Physical Directory: D:\dna\Misc\scottgu\debug\

Requests to this Application					Remaining: 6
No.	Request Time	File	Status	Verb	
1	06/21/2000 14:44:18	/AspConf.aspx	200	POST	<a href="#">View Details</a>
3	06/21/2000 14:44:45	/AspConf.aspx	200	POST	<a href="#">View Details</a>
4	06/21/2000 14:44:47	/AspConf.aspx	200	POST	<a href="#">View Details</a>

Done Local intranet

# Debugging

- **Basic Debugger ships with the SDK**
  - Multi-language
  - Single stack trace
  - Breaks, watches, etc.
- **Visual Studio® Debugger**
  - Adds remote debugging support
  - Supports managed/unmanaged debugging



# Debugging Steps

1. Create “config.web” file in app

**root:**

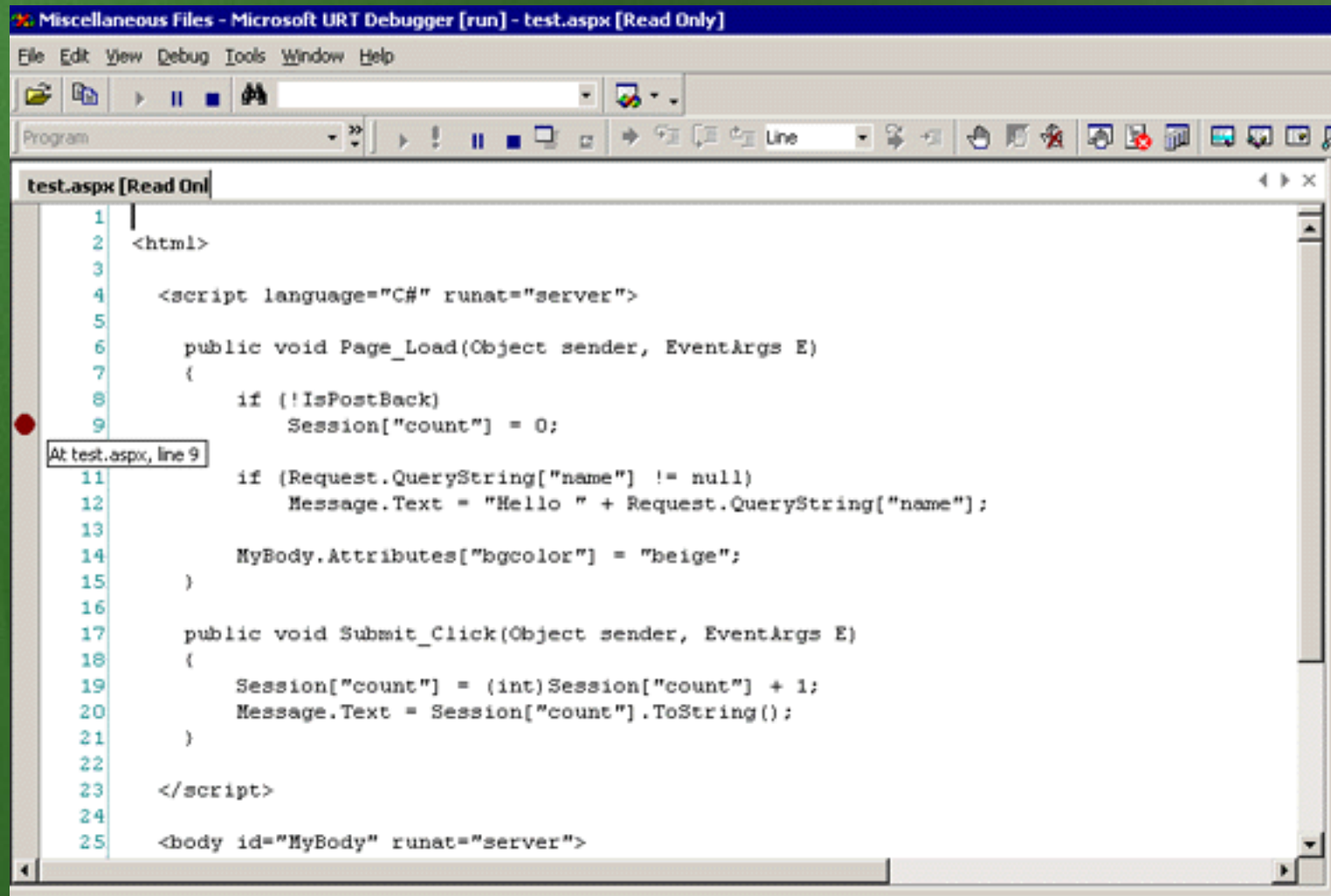
```
<configuration>
```

```
  <compilation debugmode=“true”/>
```

```
</configuration>
```

2. Attach using debugger
3. Set breakpoints
4. Hit page or service

# Debugger Demo



The screenshot shows the Microsoft URT Debugger interface. The title bar reads "Miscellaneous Files - Microsoft URT Debugger [run] - test.aspx [Read Only]". The menu bar includes File, Edit, View, Debug, Tools, Window, and Help. A toolbar with various icons is located below the menu. The main window displays the source code of test.aspx, which is a web page with a C# script block. A red dot on the left margin indicates a breakpoint set at line 9. A tooltip above the breakpoint reads "At test.aspx, line 9". The code is as follows:

```
1 |
2 <html>
3
4 <script language="C#" runat="server">
5
6     public void Page_Load(Object sender, EventArgs E)
7     {
8         if (!IsPostBack)
9             Session["count"] = 0;
10
11         if (Request.QueryString["name"] != null)
12             Message.Text = "Hello " + Request.QueryString["name"];
13
14         MyBody.Attributes["bgcolor"] = "beige";
15     }
16
17     public void Submit_Click(Object sender, EventArgs E)
18     {
19         Session["count"] = (int)Session["count"] + 1;
20         Message.Text = Session["count"].ToString();
21     }
22
23 </script>
24
25 <body id="MyBody" runat="server">
```

# Error Handling

- **.Net Common Language Runtime provides unified exception architecture**
  - Runtime errors done using exceptions
  - VB now supports try/catch/finally
- **ASP+ also provides declarative application custom error handling**
  - Automatically redirect users to error page when unhandled exceptions occur

# Custom Error Page Steps

1. Create “config.web” file in app

root:

```
<configuration>
```

```
    <customerrors mode=“remoteonly”  
    defaultredirect=“error.htm”>
```

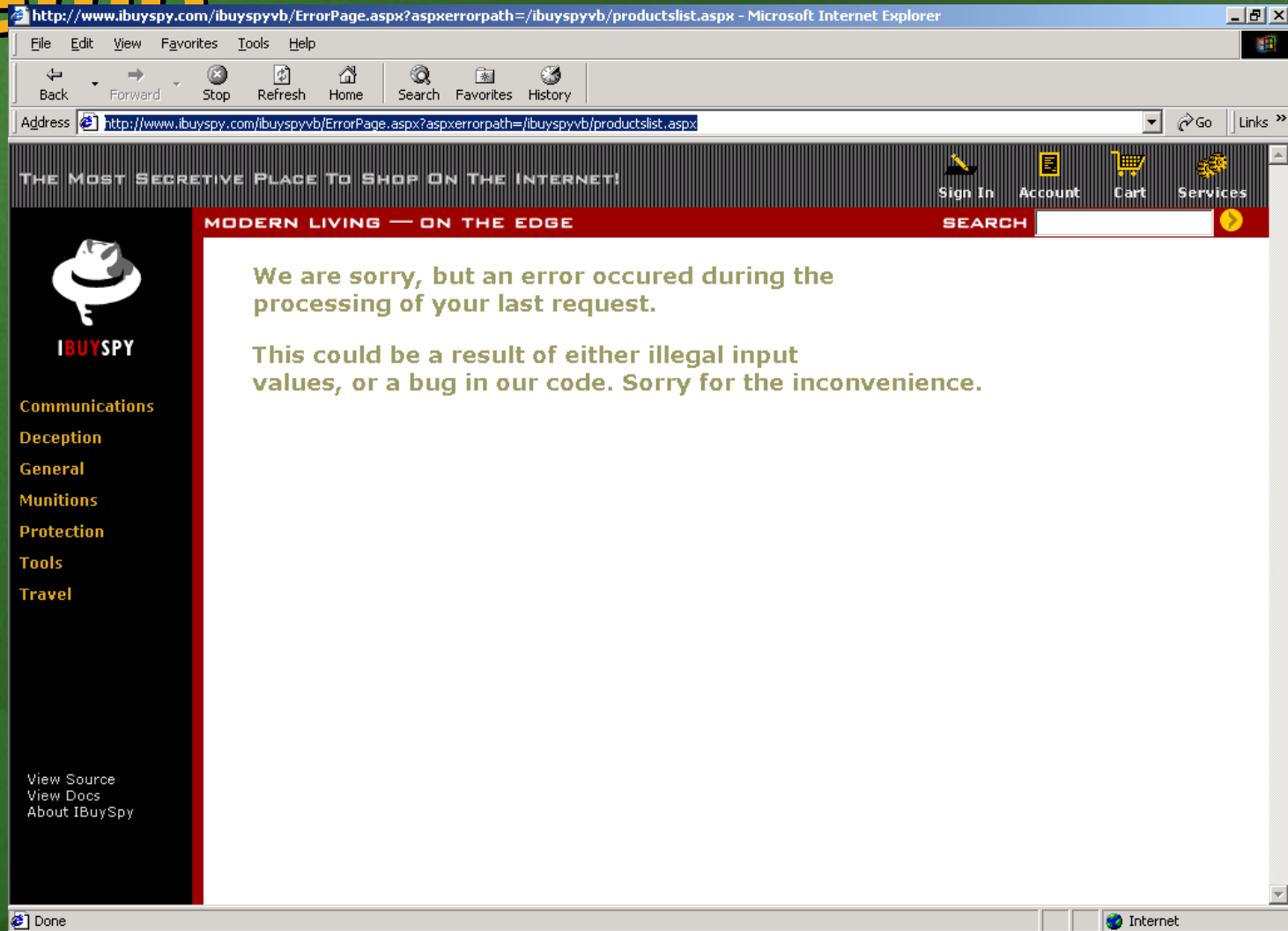
```
        <error statusCode=“404”  
        redirect=“adminmessage.htm”/>
```

```
        <error statusCode=“403”  
        redirect=“noaccessallowed.htm”/>
```

```
    </customerrors>
```

```
</configuration>
```

# Custom Error Pages Demo



# **Application\_Error**

- **Global application event raised if unhandled exception occurs**
  - Provides access to current Request
  - Provides access to Exception object
- **Cool Tip:**
  - Use new **EventLog** class to write custom events to log when errors occur
  - Use new **Smtplib** class to send email to administrators



# Writing to the NT Event Log

```
<%@ Import Namespace="System.Diagnostics" %>  
<%@ Assembly name="System.Diagnostics" %>
```

```
<script language="VB" runat=server>
```

```
Public Sub Application_Error(Sender as Object, E as EventArgs)
```

```
    Dim LogName As String = "MyCustomAppLog"
```

```
    Dim Message As String = "Url " & Request.Path & " Error: " &  
Me.Error.ToString()
```

```
    ' Create Event Log if It Doesn't Exist
```

```
    If (Not EventLog.SourceExists(LogName)) Then
```

```
        EventLog.CreateEventSource(LogName, LogName)
```

```
    End if
```

```
    ' Fire off to Event Log
```

```
    Dim Log as New EventLog
```

```
    Log.Source = LogName
```

```
    Log.WriteEntry(Message, EventLogEntryType.Error)
```

```
End Sub
```

```
</script>
```

# Sending SMTP Mail

```
<%@ Import Namespace="System.Web.Util" %>
```

```
<script language="VB" runat=server>
```

```
Public Sub Application_Error(Sender as Object, E as EventArgs)
```

```
Dim MyMessage as New MailMessage
```

```
MyMessage.To = "scottgu@microsoft.com"
```

```
MyMessage.From = "MyAppServer"
```

```
MyMessage.Subject = "Unhandled Error!!!"
```

```
MyMessage.BodyFormat = MailFormat.Html
```

```
MyMessage.Body = "<html><body><h1>" & Request.Path &  
"</h1>" &
```

```
Me.Error.ToString() & "</body></html>"
```

```
SmtptMail.Send(MyMessage)
```

```
End Sub
```

```
</script>
```

# **Running Web Applications**

The background is a solid green color. In the lower-left quadrant, there are three faint, semi-transparent green circles of varying sizes. Thin, light-green lines connect these circles, forming a network-like pattern that extends towards the center of the slide.

# Process Model

## Recovery

- ASP+ runs code in an external worker process - xspwp.exe
  - Automatic AV/Crash Protection
- It is also now possible to configure ASP+ worker process to recover from:
  - Memory Leaks
  - Deadlocks

# Handling Deadlocks

## 1. Open root system configuration file

- C:\WIN2000\ComPlus\ v2000.14.1812

## 2. Edit `<iisprocessmodel>` settings:

```
<iisprocessmodel>  
  requestqueue limit="500"  
</iisprocessmodel>
```

# Handling Memory Leaks

1. Open root system configuration file

□ C:\WIN2000\ComPlus\  
v2000.14.1812

2. Edit `<iisprocessmodel>`  
settings:  
memorylimit="75"  
/>



# Process Model Demo

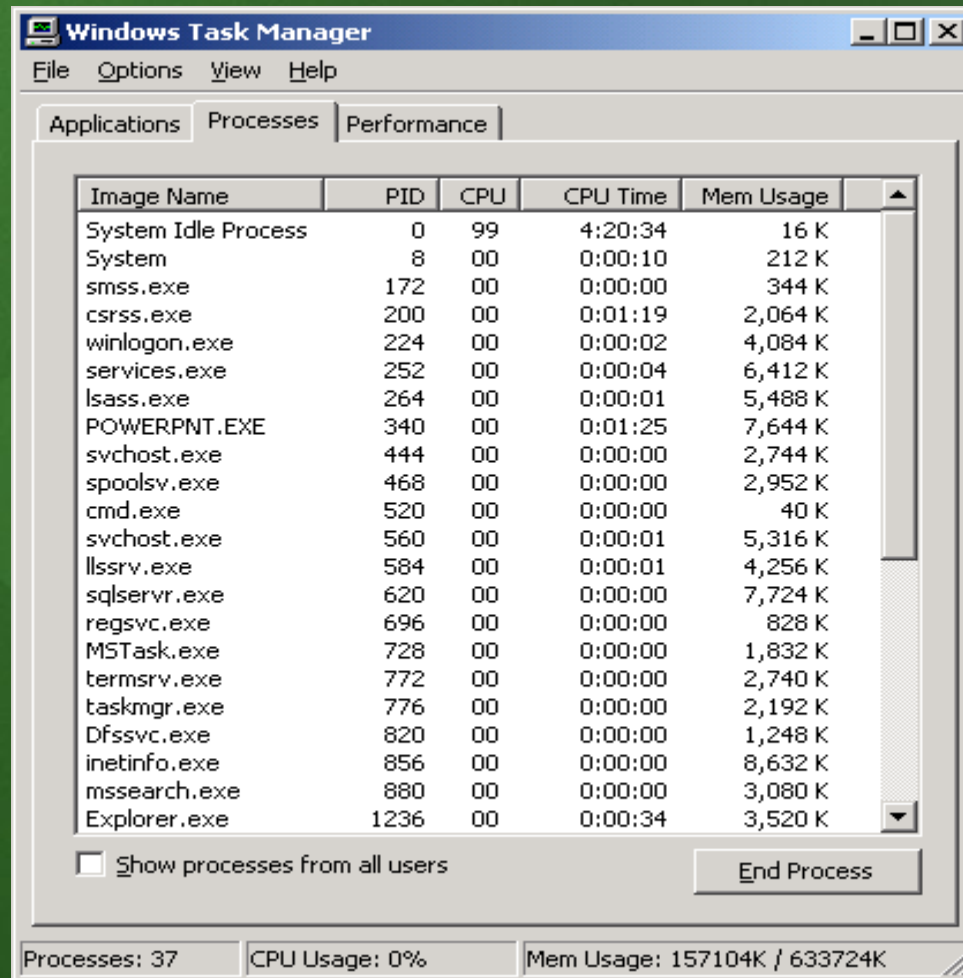


Image Name	PID	CPU	CPU Time	Mem Usage
System Idle Process	0	99	4:20:34	16 K
System	8	00	0:00:10	212 K
smss.exe	172	00	0:00:00	344 K
csrss.exe	200	00	0:01:19	2,064 K
winlogon.exe	224	00	0:00:02	4,084 K
services.exe	252	00	0:00:04	6,412 K
lsass.exe	264	00	0:00:01	5,488 K
POWERPNT.EXE	340	00	0:01:25	7,644 K
svchost.exe	444	00	0:00:00	2,744 K
spoolsv.exe	468	00	0:00:00	2,952 K
cmd.exe	520	00	0:00:00	40 K
svchost.exe	560	00	0:00:01	5,316 K
lsrv.exe	584	00	0:00:01	4,256 K
sqlservr.exe	620	00	0:00:00	7,724 K
regsvc.exe	696	00	0:00:00	828 K
MSTask.exe	728	00	0:00:00	1,832 K
termsrv.exe	772	00	0:00:00	2,740 K
taskmgr.exe	776	00	0:00:00	2,192 K
Dfssvc.exe	820	00	0:00:00	1,248 K
inetinfo.exe	856	00	0:00:00	8,632 K
mssearch.exe	880	00	0:00:00	3,080 K
Explorer.exe	1236	00	0:00:34	3,520 K

☐ Show processes from all users

End Process

Processes: 37   CPU Usage: 0%   Mem Usage: 157104K / 633724K

# Pre-emptive cycling

- **ASP+ optionally supports pre-emptive cycling of worker processes**
  - Eliminates need for admins to “kick the server” once a week
- **Can be configured two ways:**
  - Time Based (reset every m minutes)
  - Request Based (reset every n requests)

# Timer Cycling

## 1. Open root system configuration file

- C:\WIN2000\ComPlus\ v2000.14.1812

## 2. Edit `<iisprocessmodel>` settings:

```
<iisprocessmodel  
  timeout="60"  
>
```

# Request Cycling

## 1. Open root system configuration file

- C:\WIN2000\ComPlus\ v2000.14.1812

## 2. Edit `<iisprocessmodel>` settings:

```
<iisprocessmodel  
requestlimit="10000"  
>
```

# Process Model Demo

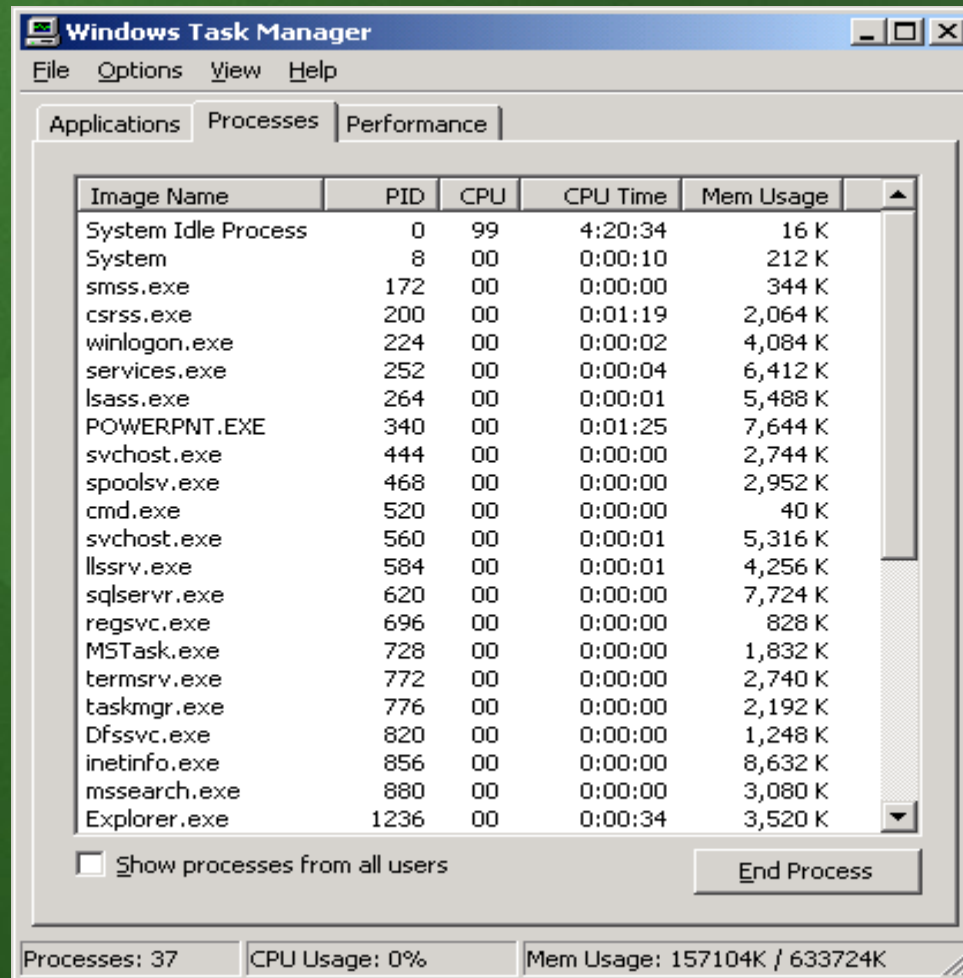


Image Name	PID	CPU	CPU Time	Mem Usage
System Idle Process	0	99	4:20:34	16 K
System	8	00	0:00:10	212 K
smss.exe	172	00	0:00:00	344 K
csrss.exe	200	00	0:01:19	2,064 K
winlogon.exe	224	00	0:00:02	4,084 K
services.exe	252	00	0:00:04	6,412 K
lsass.exe	264	00	0:00:01	5,488 K
POWERPNT.EXE	340	00	0:01:25	7,644 K
svchost.exe	444	00	0:00:00	2,744 K
spoolsv.exe	468	00	0:00:00	2,952 K
cmd.exe	520	00	0:00:00	40 K
svchost.exe	560	00	0:00:01	5,316 K
lsrv.exe	584	00	0:00:01	4,256 K
sqlservr.exe	620	00	0:00:00	7,724 K
regsvc.exe	696	00	0:00:00	828 K
MSTask.exe	728	00	0:00:00	1,832 K
termsrv.exe	772	00	0:00:00	2,740 K
taskmgr.exe	776	00	0:00:00	2,192 K
Dfssvc.exe	820	00	0:00:00	1,248 K
inetinfo.exe	856	00	0:00:00	8,632 K
mssearch.exe	880	00	0:00:00	3,080 K
Explorer.exe	1236	00	0:00:34	3,520 K

☐ Show processes from all users      End Process

Processes: 37      CPU Usage: 0%      Mem Usage: 157104K / 633724K

# Persistent Compiles

- **ASP+ saves dynamic compilations (.ASPX, .ASMX) to disk**
  - **Avoids recompile on process startup**
  - **No special tool or instructions required**
- **Should dramatically improve startup time on large applications**
- **Caveat:**



# Reliable Session State

- **Session State can now be external from ASP+ Worker Process**
  - ASPState Windows NT Service
  - SQL Server™
- **Big reliability wins**
  - Session state survives crashes/restarts
- **Enables Web farm deployment**
  - Multiple FE machines point to a common state store

# External Session State Steps

1. Start ASP State Service on a machine

□ net start aspstate

2. Create “config.web” file in app vroot, and point it at state

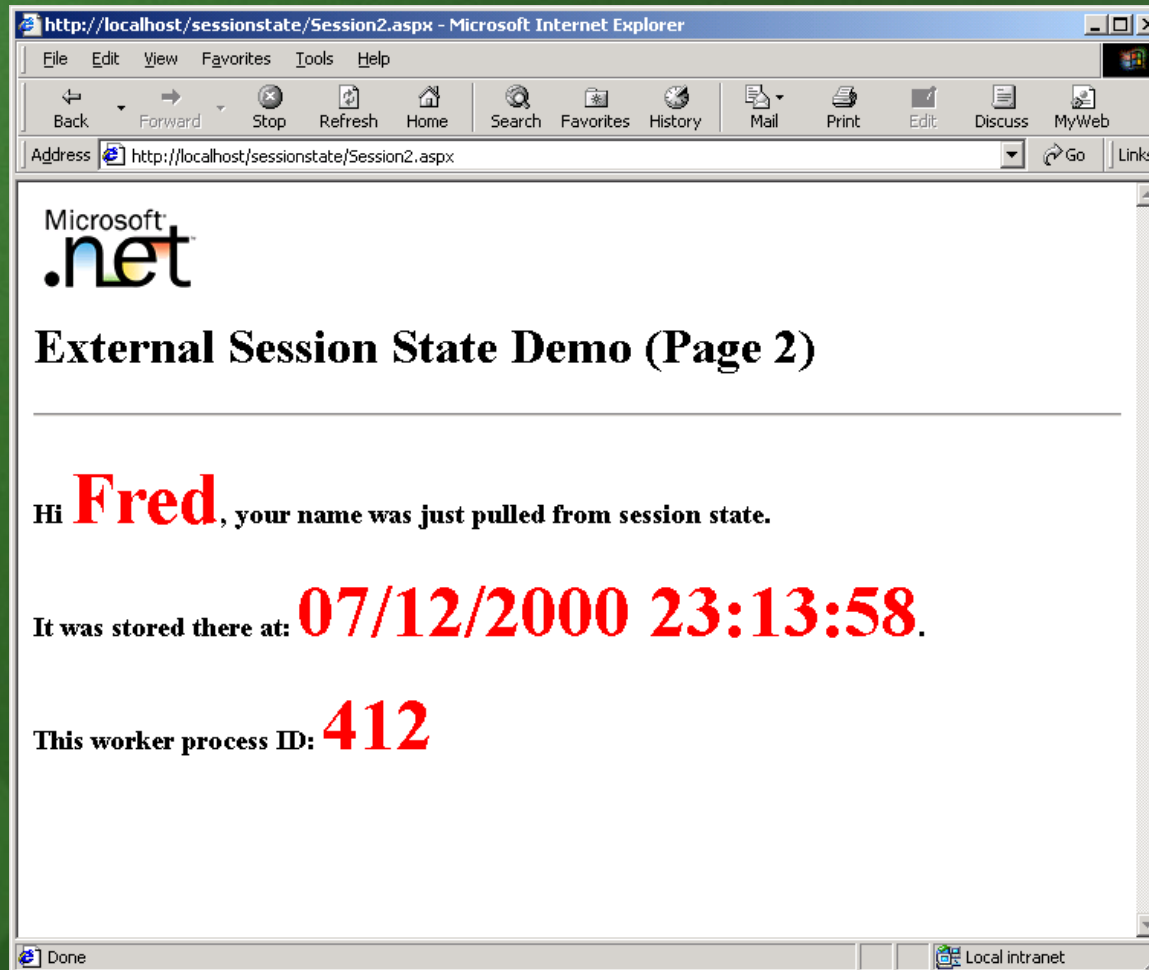
<configuration>

service machine:

<sessionstate inproc=“false” server=“localhost” />

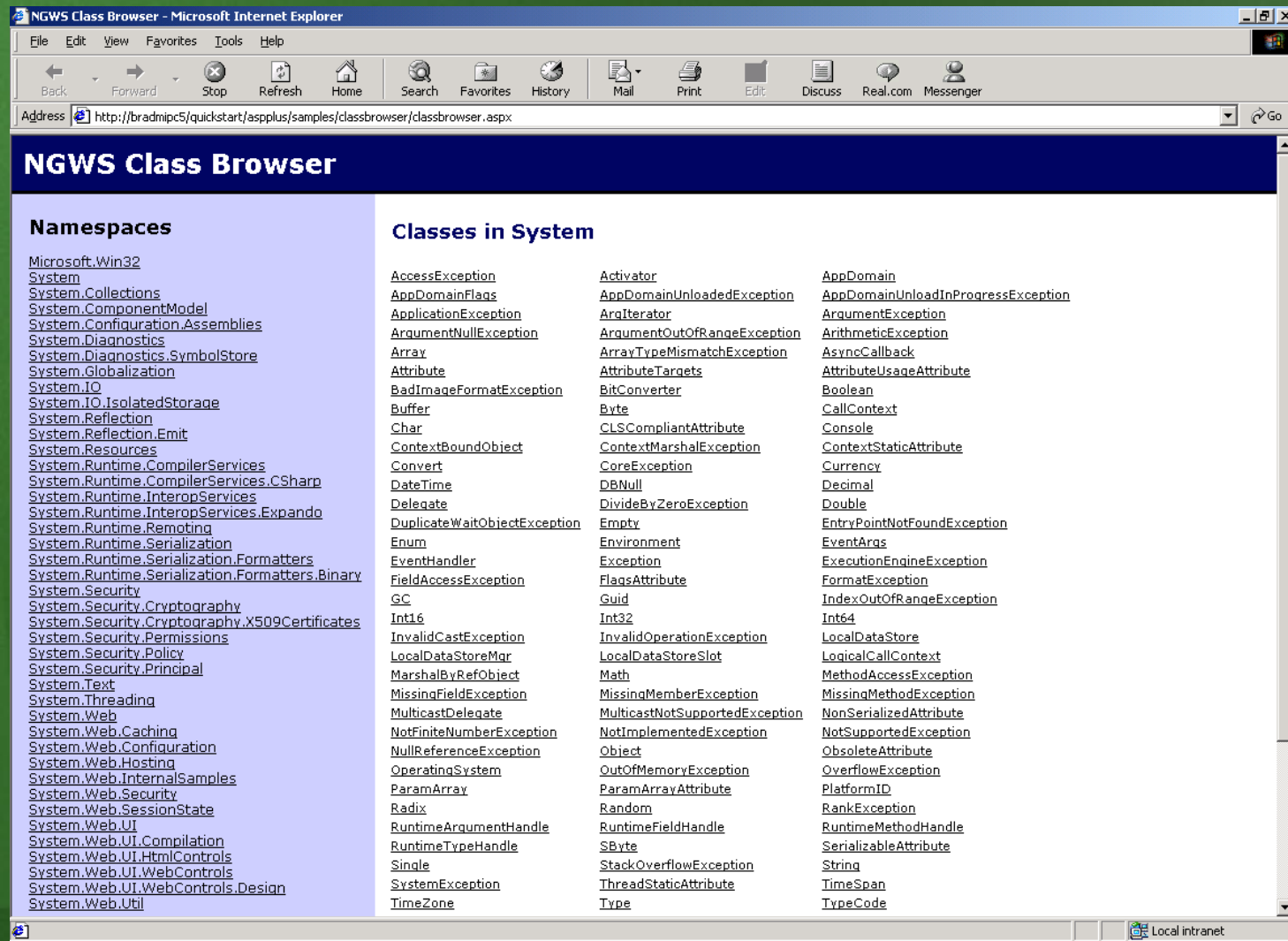
</configuration>

# External Session State Demo



# **Miscellaneous Other Fun Things**

# ClassBrowser



# **File Upload**

- **ASP+ provides file upload support**
  - **Request.Files collection**
  - **<input type=file> server control**
- **HttpPostedFile class**
  - **HttpPostedFile.InputStream**
  - **HttpPostedFile.ContentLength**
  - **HttpPostedFile.ContentType**
  - **HttpPostedFile.FileName**
  - **HttpPostedFile.SaveAs(fileLocation)**



# File Upload Example

```
<html>
```

```
  <script language="VB" runat=server>
```

```
    Sub UploadBtn_Click(Sender as Object, E as EventArgs)  
      UploadFile.PostedFile.SaveAs("c:\foo.txt")  
    End Sub
```

```
</script>
```

```
<body>
```

```
  <form enctype="multipart/form-data" runat=server>
```

```
    Select File To Upload: <input id="UploadFile" type=file  
runat=server>
```

```
    <asp:button Text="Upload!" OnClick="UploadBtn_Click"  
runat=server/>
```

```
</form>
```

```
</body>
```

# Image Generation

- **System.Drawing can perform rich server image generation/manipulation**
  - **Image Generation**
  - **Overlays on top of existing images**
  - **Resizing/Cropping images**
  - **Read/Write Any Standard IO Stream**

# **Image Generation Demo**

# Summary

- **Hundreds of cool features to use when building ASP+ Web Applications**
  - Only a small sampling shown here...
- **For more information/samples please review the ASP+ Quickstart**
  - 400+ samples that can be run online
- **For a good “best practices” reference application, please**

# **Other Information**

## **Sources**

- **Please review the ASP+ hands on tutorial**
  - **Photo album sample walkthrough**
- **Public ASP+ Community Sites:**
  - **[www.learnasp.com](http://www.learnasp.com)**
  - **[www.aspng.com](http://www.aspng.com)**
  - **[www.asp101.com](http://www.asp101.com)**
  - **[www.aspfree.com](http://www.aspfree.com)**

# Server Side Comments

- **ASP+ supports server-side comments**
  - **<%-- Omit Me --%>**
  - **Useful development tool**

- **Example:**

**<%--**

**Some random text....**

**<asp:calendar id="MyCal"  
runat=server/>**

**<%= Now %>**

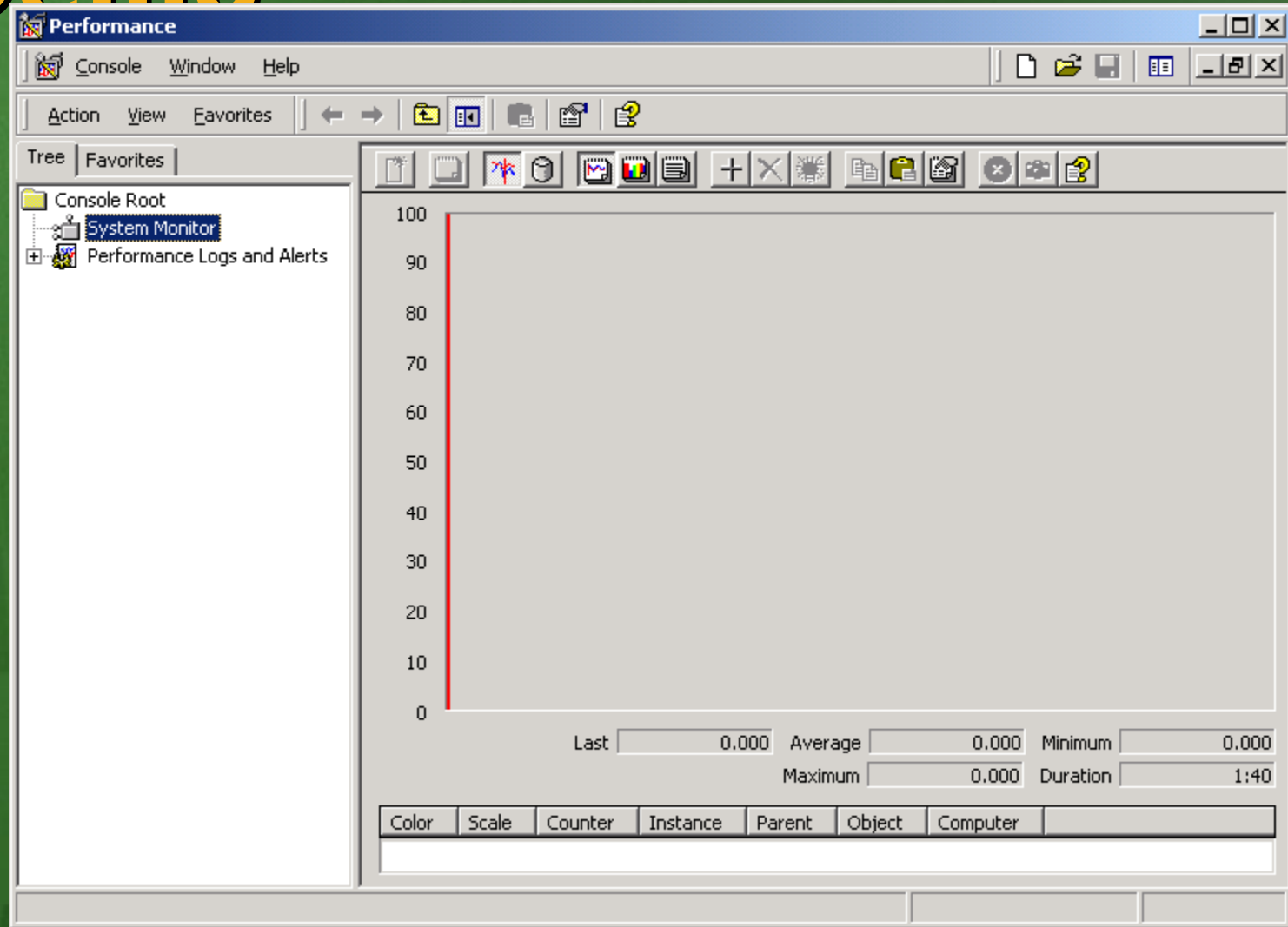
**--%>**



# Performance Counters

- **Per Application Performance Counters**
  - Now possible to easily identify how individual applications are functioning
- **Custom Performance Counters APIs**
  - Now possible to publish unique app-specific performance data
  - Example: total orders, orders/sec

# Performance Counter Demo



# Discussion



Where do **you** want to go today?

**Microsoft**